Analogical Explanations

Vibhu O. Mittal and Cecile L. Paris
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, California 90292

December 1990
ISI/RR-93-317

INFORMATION SCIENCES INSTITUTE

310/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

Analogical Explanations

Vibhu O. Mittal and Cecile L. Paris
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, California 90292

December 1990
ISI/RR-93-317

In the Proceedings of the Third Conference on Knowledge Based Computer Systems (KBSC-90)

REPORT DOCUMENTATION PAGE	*FORM APPROVED OMB NO. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE December 1990	3. REPORT TYPE AND DATES COVERED Research Report

4. TITLE AND SUBTITLE Analogical Explanations	5. FUNDING NUMBERS NCC2-250
6. AUTHOR(S) Vibhu O. Mittal and Cecile Paris	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) USC INFORMATION SCIENCES INSTITUTE 4676 ADMIRALTY WAY MARINA DEL REY, CA 90292-6695	8. PERFORMING ORGANIZATON REPORT NUMBER RR-317
9. SPONSORING/MONITORING AGENCY NAMES(S) AND ADDRESS(ES) NASA AMES Moffett Field, CA 94035	10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

In Proceedings of the Third Conference on Knowledge Based Computer Systems - KBSC90

12A. DISTRIBUTION/AVAILABILITY STATEMENT UNCLASSIFIED/UNLIMITED	12B. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Analogical problem solving has been an area of considerable research in AI for some time, but the use of analogies in language generation as a means of explanation has attracted much less attention. Analogies are very interesting linguistic devices, because they convey a large amount of information in a relatively concise fashion. While using analogies is very common in language, most generations systems are not able to use analogies when providing explanations. In this paper, we discuss analogies in general, and outline a framework which will lay the foundations for knowledge representation suitable for generating inter-domain analogies. We argue that language should help in knowledge representation, and employ the Upper-Model, an ontology of abstractions used in natural language generation, to help organise representation of domain terminology.

14. SUBJECT TERMS explanations, analogy, natural language generation	15. NUMBER OF PAGES 10
	16. PRICE CODE

17. SECURITY CLASSIFICTION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

GENERAL INSTRUCTIONS FOR COMPLETING SF 298

The Report Documentation Page (RDP) is used in announcing and cataloging reoprts. It is important that this information be consistent with the rest of the report, particularly the cover and title page. Instructions for filling in each block of the form follow. It is important to stay within the lines to meet optical scanning requirements.

Block 1. Agency Use Only (Leave blank).

Block 2. Report Date. Full publication date including day, month,a nd year, if available (e.g. 1 jan 88). Must cite at least the year.

Block 3. Type of Report and Dates Covered. State whether report is interim, final, etc. If applicable, enter inclusive report dates (e.g. 10 Jun 87 - 30 Jun 88).

Block 4. Title and Subtitle. A title is taken from the part of the report that provides the most meaningful and complete information. When a report is prepared in more than one volume, repeat the primary title, add volume number, and include subtitle for the specific volume. On classified documents enter the title classification in parentheses.

Block 5. Funding Numbers. To include contract and grant numbers; may include program element numbers(s), project number(s), task number(s), and work unit number(s). Use the following labels:

C	- Contract	PR	- Project
G	- Grant	TA	- Task
PE	- Program Element	WU	- Work Unit Accession No.

Block 6. Author(s). Name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. If editor or compiler, this should follow the name(s).

Block 7. Performing Organization Name(s) and Address(es). Self-explanatory.

Block 8. Performing Organization Report Number. Enter the unique alphanumeric report number(s) assigned by the organization performing the repor.

Block 9. Sponsoring/Monitoring Agency Names(s) and Address(es). Self-explanatory

Block 10. Sponsoring/Monitoring Agency Report Number. (If known)

Block 11. Supplementary Notes. Enter information not included elsewhere such as: Prepared in cooperation with...; Trans. of ...; To be published in... When a report is revised, include a statement whether the new report supersedes or supplements the older report.

Block 12a. Distribution/Availability Statement. Denote [illegible] c availability or limitations. Cite any availa [illegible] he public. Enter additional limitat [illegible] pecial markings in all capitals (e.g. NOFORN, [illegible], ITAR).

DOD	- See DoDD 5230.24, "Distribution Statements on Technical Documents."
DOE	- See authorities.
NASA	- See Handbook NHB 2200.2.
NTIS	- Leave blank.

Block 12b. Distribution Code.

DOD	- Leave blank.
DOE	- Enter DOE distribution categories from the Standard Distribution for Unclassified Scientific and Technical Reports.
NASA	- Leave blank.
NTIS	- Leave blank.

Block 13. Abstract. Include a brief (Maximum 200 words) factual summary of the most significant information contained in the report.

Block 14. Subject Terms. Keywords or phrases identifying major subjects in the report.

Block 15. Number of Pages. Enter the total number of pages.

Block 16. Price Code. Enter appropriate price code (NTIS only).

Blocks 17.-19. Security Classifications. Self-explanatory. Enter U.S. Security Classification in accordance with U.S. Security Regulations (i.e., UNCLASSIFIED). If form contins classified information, stamp classification on the top and bottom of the page.

Block 20. Limitation of Abstract. This block must be completed to assign a limitation to the abstract. Enter either UL (unlimited) or SAR (same as report). An entry in this block is necessary if the abstract is to be limited. If blank, the abstract is assumed to be unlimited.

Analogical Explanations

Vibhu O. Mittal & Cécile L. Paris

USC/Information Sciences Institute
4676 Admiralty Way
Marina del Ray, CA 90292
U.S.A.
mittal@vaxa.isi.edu paris@vaxa.isi.edu

Abstract

Analogical problem solving has been an area of considerable research in AI for some time, but the use of analogies in language generation as a means of explanation has attracted much less attention. Analogies are very interesting linguistic devices, because they convey a large amount of information in a relatively concise fashion. While using analogies is very common in language, most generation systems are not able to use analogies when providing explanations. In this paper, we discuss analogies in general, and outline a framework which will lay the foundations for knowledge representation suitable for generating inter-domain analogies. We argue that language should help in knowledge representation, and employ the *Upper-Model*, an ontology of abstractions used in natural language generation, to help organise the representation of domain terminology.

1 Introduction

Explanation is a very important problem solving activity in its own right, and much work has been conducted in this area of research. However, generating analogies to explain concepts seems to have been over-looked as a valuable strategy[1] to convey the meaning concisely and effectively to the user. Analogical explanation is particularly useful in explaining concepts to a user in terms of the concepts in another domain that the user is familiar with. Consider for example the dialog in figure 1.

In this hypothetical dialog, the analogy provides an effective strategy for explaining the concept of thrombolysis. Since we are doing this work in the context of explaining an expert system's knowledge base and reasoning, we are considering using analogies in the following two cases:

1. In describing a concept or a relation – which is an essential capability of any explanatory system. Indeed, a svstem must be able to explain terms a user does

[1]Some generation systems are able to provide some analogies to describe concepts, but this is done in a very limited sense of analogy and for a limited number of cases.

"What is Thrombolysis?"
"**Thrombolysis is lysing a blood clot by infusing an enzyme in the blood**" (Normal explanation)
"Huh?" (confused reaction)
"**Just as one dissolves a blockage in a pipe by adding 'Draino', one can dissolve a blood clot by adding an enzyme**" (analogical explanation)

Figure 1: A Hypothetical Dialog illustrating the use of analogies in Explanation

not understand. An example of the use of an analogy in the explanation of a terminological definition is:

> The mitral valve in the heart is like a diode in an electronic circuit
> (They both allow passage in one direction only)

2. In explaining a series of actions performed – as in a problem solving trace. For example:

> Catheterisation is first performed on the patient. This is a diagnostic procedure similar to the one where a network of pipes is diagnosed to find the blockage. On finding that an artery is blocked partially by a thrombus, the patient is injected with enzymes so as to lyse the thrombus and restore normal circulation. This is like injecting a pipe with a strong acidic solution to dissolve the solid blocking the pipe and restoring normal water flow in the network.

These explanations must not be spewed from pre-written canned pieces of text, but must instead be generated by perusing the knowledge base. Thus if the system designer changes the definition of a concept, or a method for solving a problem, the explanation generated by the system should also change appropriately automatically. Equally important, this form of knowledge representation supports the generation of different explanations in different situations, based on different user models (e.g., [Paris, 1989; Paris, 1990]). In all cases, the explanatory capabilities of a system would be greatly enhanced if it could also make use of analogies.

Before discussing how one might generate analogies, it is important to first distinguish an *analogy* from either a *metaphor* or a *comparison*. When two very similar items are contrasted or compared, as in the following statements:

1. An artery is like a vein
2. An LED is like a bulb

we are really making comparisons rather than drawing analogies. The two entities being discussed are usually from the same immediate category (that is, they share the same parent in the knowledge representation hierarchy). However, when we make statements such as:

3. The roads of a city are like the veins and arteries of a body
4. He runs like the wind

we are drawing analogies between two dissimilar entities. An analogy is drawn between two concepts of different *categories*, and is therefore more than a mere comparison of two concepts. Generation of an analogy requires identifying the similarities between features of the two concepts – in the case above for example, the need for transport channels and the speed of movement and disregarding the other surface-level features (in the case above, the fact that the wind is not a solid for example). Systems that have used analogies in generation in the past have dealt only with comparisons such as (1) and (2).

Metaphors are yet even more complex and we shall mention them only for completeness. They usually leave implicit what the analogy explicitly mentions, and they require a context to be interpreted in.[2] For example:

5. The roads are its arteries and veins
6. He is the wind

Since analogies draw upon the similarities between the features of two objects in different categories, they would seem to be a logical candidate for the transfer of knowledge about a domain in terms of another domain. Most of the work in analogical reasoning has been in the transfer of knowledge *to* the system (i.e., for knowledge acquisition). There has been little or no work done in the transfer of knowledge *from* the system to the user, as for example, in explanation.

In the following sections, we will describe the skeleton of a framework, which will lay the foundations for a knowledge representation suitable for generating analogies.

2 Analogies in Problem Solving *vs* Explanation Generation

In problem solving, analogies have been used mainly in order to:

1. To help the system solve a new problem based on its properties which are analogous to some other previously solved problem (e.g. [Greiner, 1985])

2. To avoid deriving a solution again, if an analogous problem trace exists (e.g., [Carbonell, 1983; Kedar-Cabelli, 1988])

There, the major problem has been the problem of *finding* the analogous concept or problem. Two of the approaches usually employed are:

- Matching the goal state and the initial state to find analogous instances.[3] Thus, for example,

  GO-FROM Los Angeles TO New York

  might be retrieved when a goal of the form GO-FROM ⟨Location-A⟩ TO ⟨Location-B⟩ is posted and a similar sequence of steps would be taken to solve this new goal, with the appropriate change of variables. This approach misses analogous situations when the goals happen to be expressed differently (or non-analogous problems may have similar goal-states) and thus is not very reliable.

[2]Some of these distinctions are discussed in greater detail in Saha [Saha, 1988].

[3]This is sometimes referred to as *Transformational Analogy*.

- Matching the *problem solving trace* instead of the initial and goal states.[4] This helps overcome the problem of missing similar problems because the were expressed differently. As long as the problem was solved in an 'analogous' manner, this approach finds a plausible analogy.

Both these approaches raise some difficult questions:

(a) How are the various steps to be 'twiddled' or 'tweaked' so that the traces match?

(b) What execution trace are we supposed to use to find an analogy when the analog is supposed to help us solve the given problem? This is generally dealt with by partial matches, which however, complicates the matching process.

There are many other problems in analogical problem solving, but that is not within the purview of this paper. In this paper, we argue that the problem of using analogies in explanation is quite a different one. In explanation, we are concerned, not with using an analogy to help *solve* the problem, but with *helping to convey* an idea.

This difference in the nature of the problem means that the questions facing analogy generation for explanation are somewhat different from the ones faced by those in problem-solving. This is because of all the knowledge available to the generator.

In particular, we will show how the EES framework provides a lot of the knowledge necessary to find an analogy, and the matching problem is thus greatly reduced. Indeed, first the terminology knowledge base provided in an expert system built using EES, gives a lot of information that can be used to generate an analogy for describing a concept. Furthermore, the problem of partial match of an execution trace is greatly simplified – in problem solving systems, the problem solver must alternate between trying to find a partial match with other traces in the library, and incrementally solve the given problem until it can find an analog. This complicates matters since the problem solver can make a wrong decision based on the incomplete information that is available to it, and must later fall back to reasoning from first principles when the analogy fails to help. This is not true in our case, since the generator can make an educated decision based on its complete knowledge about *both* problem solving traces.

We shall now review the beginnings of an environment in which we believe this can be done.

3 The EES system and its explanation facility

The Explainable Expert System (EES) framework provides a shell in which to build expert systems. In this framework, an expert system includes much support knowledge for explanation [Neches *et al.*, 1985; Swartout and Smoliar, 1987; Moore, 1989]. The support knowledge contains, for example, terminology definitions, so that a user can ask for the definition of specific terms used by the system, problem-solving principles and problem solving methods. Due to the lack of space, we will not discuss architecture of the EES framework here. The interested reader is referred to the articles mentioned above.

The EES system is built using the LOOM [MacGregor, 1988] knowledge representation language, and makes use of the PENMAN Upper-Model [Bateman *et al.*, 1989], a knowledge base of general conceptual categories. The Upper-Model is a computational resource for organizing domain knowledge appropriately for linguistic realizations. It

[4]This approach is sometimes also referred to as *Derivational Analogy*.

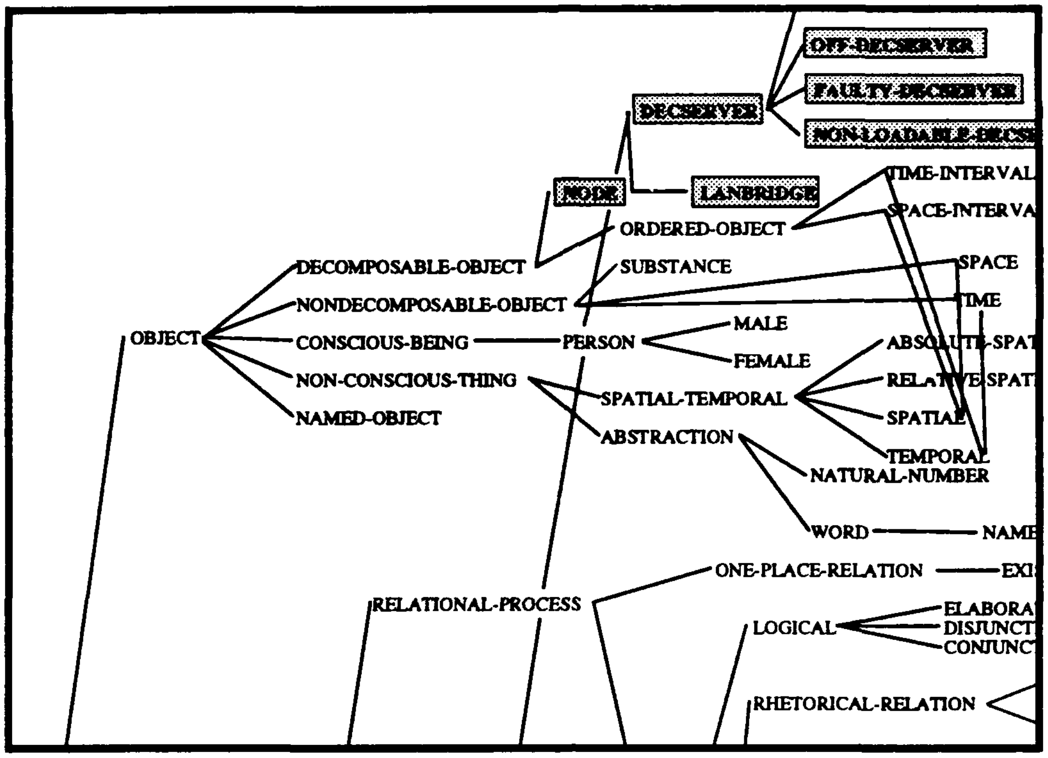

Figure 2: A portion of the Upper-Model concept hierarchy

provides a domain- and task-independent classification system that supports sophisticated natural language processing. The domain knowledge for a particular application is linked into the Upper-Model. A small section of the concept hierarchy in the Upper-Model is shown in Figure 2 to illustrate how the LANALYST (the local-area networks) domain was defined underneath the Upper-Model. (The LANALYST concepts are shown highlighted in the figure.)

Since the Upper-Model is domain independent, any knowledge base can be built under it. This feature allows us to employ the Upper-Model to generate analogies between concepts in different knowledge bases by using the Upper-Model as a bridge between the different knowledge bases.

4 Description of the Model

We are currently building a prototype expert system in the EES framework to do local area networks analysis. We are also collaborating with the cardiology group at the Cedars-Sinai Hospital to construct a system to diagnose and suggest management plans for post-myocardial infarctions. We will be testing explanation by analogy in those two systems – however for the purposes of clarification, we will use examples from everyday life in this paper.

Explanations in our system are either terminological ones – i.e., they explain terms in our knowledge base, either as a result of an explicit request, or while explaining the reasoning of the system; or they can be explanations that are meant to explain and justify why certain actions were taken and decisions made – essentially an explanation of the problem solving activity.

4.1 Terminological Analogies

There can be two types of analogies that the system can draw when it needs to explain the concept more fully:

4.1.1 Structural Analogies

These analogies are of the form:

> A plate is made of china, just as a cup is made of glass.

When called upon to explain a point (the material an object is made of, in the above example), the system draws such analogies first. These analogies can be made by carefully traversing the concept model hierarchy, and using principles similar to those in Gentner's Structure Mapping Engine [Gentner, 1983]. In drawing this analogy, the system needs to traverse its hierarchy of abstractions and match similarities in the concept descriptions – in terms of the parent concepts and the relations between this concept and other concepts. To match these concepts, we need to have a set of common abstractions – for which we use the Upper-Model.

The Upper-Model essentially provides an abstract semantic hierarchy of concepts which partially constrains the linguistic realization options that may be considered by the grammar. Concepts from particular domains that are to be expressed in English are subordinated to concepts in the Upper-Model: their placement in the Upper-Model then determines the range of realizational possibilities that those concepts have.[5] For example, a 'directed-action' in the Upper-Model is a particular subtype of 'action' that requires two participants, an 'actor' and an 'actee', which in turn may have particular semantic constraints on their fillers. By classifying a domain concept under 'directed-action', we indicate to the grammar how to realize this concept in English. The important point for us is that the Upper-Model provides us with a *domain independent abstract semantic hierarchy*. As a result, completely unrelated knowledge bases may be defined underneath it, and our system will be able to draw analogies between concepts in different knowledge bases. As a result of this exercise, we have also come to the realization that a more domain-specific layer below the Upper-Model would be required to constrain the searches for analogues in larger knowledge bases.

In deriving an analogy for a structural terminological definition, our system would try to match only those relations that "interact" with other concepts. Independent or unary relations are matched after binary relations, so that "irrelevant" properties may not affect the outcome as much as relations between concepts do.

4.1.2 Functional Analogies

Another way to draw an analogy is to find an object with a similar 'purpose'. Consider for example, a paper cup. A paper cup can be used in many ways, but if we are using it to drink water, and we need to find an analogous concept. The property of the cup that we are most concerned with is the ability to drink from it. This suggests a way

[5]For full details of this process, see [Bateman *et al.*, 1989; The Penman Natural Language Group, 1989].

of focusing the match process. This is important, because quite often, matching all the roles of a concept with those of another concept is not a trivial task, since the number of relations defined on the concept may be very large. Consider for example, the concept Decserver in our LANALYST domain which has 14 roles:

|R|SOURCE-COMPUTER |R|D1 |R|D2 |R|D3 |R|D4 |R|COMPONENT-SYMPTOM
|R|CONNECTION |R|COMPONENT-STATUS |R|POWER-STATE
|R|INDICATOR-STATUS |R|LOAD-STATUS |R|EXHIBITS-SYMPTOM-CAUSE
|R|NETWORK-CONNECTION |R|DECSERVER-EXPECTED-VERSION

We are unlikely to need all these 14 roles to describe the DecServer at any time, and, to find an analogy, we will need only a few at a time. In addition to the relations above which were defined locally, there are also inherited relations. If all these relations are also considered, then the total number of relations to scrutinize becomes very large and the process of matching could become a difficult task.

Instead, we can use the knowledge of the 'purpose' of the object in the current context to prune the number of relations we need to match. In the case of the paper cup, for instance, we would need to find another object that could fulfill the role of the paper cup. Assuming that we were using the cup to drink water, any other object that could be used for drinking water would suit our purposes for analogy. Even though the concept "cup" may have a large number of properties defined with it (light, conical, flammable, crushable, cheap, white, etc), we need be only concerned with the properties of the cup being light (so that it is liftable) and conical (so that it can hold water). This is similar to Cabelli's purpose-directed analogy approach [Kedar-Cabelli, 1988]. This dramatically reduces the number of roles that need to be matched to find an analog.

Because the EES knowledge bases are constructed in a fashion such that information necessary for explanation is readily available and because they are linked underneath the Upper-Model, the system is able to access information such as the purpose of the object and use it to look for analogies. The Upper-Model helps us in making further inferences about the concepts and relations that are defined with respect to each other, by providing us with a common vocabulary, and match their terms and their relevant roles.

EES also represents all problem solving knowledge, both general purpose and domain dependent, in the form of explainable plans in a plan knowledge base. This gives us another way of generating functional analogies. A concept to which a particular plan is applicable, may be functionally analogous to another concept, in the context of this plan's goal, if this plan is equally applicable to the other concept. Take for example a plan which can achieve the following goal:

```
(DETERMINE-WHETHER (OBJ (X ISA COMPONENT))
                   (PROBLEM (X IS NON-CONDUCTING)))
```

and "NON-CONDUCTING" is defined as something that does not allow the passage of material from one input of X to the other. This plan is applicable to any component which has two inputs – for example a LanBridge (does not allow passage of messages from one ethernet to another) or to a pipe or an artery. In this case, a functional analogy can be drawn between a LanBridge and a Pipe when the current goal of the problem solving happens to be the one above.

4.2 Analogies based on Problem Solving Traces

Apart from explaining individual terms that the user may not understand, the EES framework is also designed to explain the plans used in solving the problem. This enables the system to explain the various decisions it took to arrive at the conclusion. Analogical explanations are desirable in this case as well and we now outline how such analogies might be generated in explaining a trace.

Suppose the system needs to explain the solution of a problem to the user and had decided to provide an analogy to a problem solving trace already explained to the user. It searches the library of traces that have already been successfully explained, and tries to find an analog problem solving trace. As noted earlier, complete traces are being stored and thus the partial match problem does not arise. Finding analogous traces is based on matching goals and concepts, thus using both LOOM and the Upper-Model and the execution trace.

Note that this raises an interesting question: Is the explanation that was produced for the analogous trace now applicable? Can, in fact, the explanation (or its design) that was kept by the system help in the matching process? i.e., a requirement might be to make sure the explanation would still be coherent, even after the appropriate variables had been changed, and possibly some steps removed. We plan to investigate these issues further.

5 Related Work

Most of the previous approaches to generating analogies have been for problem solving and not for language generation. Some work done by Brown et al. [Brown and Burton, 1975] have concentrated on the *sorts* of analogies that are useful in conveying ideas - e.g., analogies that talk in terms of connections between physical parts, or some that discuss causal effects. But no system has addressed the issue of generating analogies between different knowledge bases for explanation. We feel that because of the constraints imposed on the knowledge representation by EES (explicitness) and the Upper-Model (expressiveness in English), knowledge bases constructed in this fashion are particularly well-suited to generating analogies for explanation.

Most of the other approaches were based either on the Schankian CD-Representation, or on similar representations and depended upon a small set of primitives to generate the appropriate abstractions to do the matching. Instead, we use the Upper-Model, a semantically-oriented specification for text-generation that is richer than most of the previous formalisms and these sets of primitives. Since analogies are used widely in language, it will be interesting to see whether the use of the Upper-Model as our abstraction hierarchy gives us any leverage or not in generating them.

References

[Bateman *et al.*, 1989] John Bateman, Bob Kasper, Johanna Moore, and Richard Whitney. *The Penman Upper Model.* USC/Information Sciences Institute, Marina del Rey, CA 90282, May 1989.

[Brown and Burton, 1975] John S. Brown and R. R. Burton. Multiple representations of knowledge for tutorial reasoning. In G. Bobrow and A. Collins, editors, *Representation and Understanding*. Academic Press, New York, 1975.

[Carbonell, 1983] Jaime G. Carbonell. Derivational analogy and its role in problem solving. In *Proceedings of AAAI-83*, pages 64 – 69, Washington, DC., 1983. American Association for Artificial Intelligence.

[Gentner, 1983] Dedre Gentner. Structure mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155 – 170, 1983.

[Greiner, 1985] Russell Greiner. *Learning by Understanding Analogies*. PhD thesis, Stanford University, September 1985.

[Kedar-Cabelli, 1988] Smadar Kedar-Cabelli. Analogy – from a unified perspective. In David H. Helman, editor, *Analogical Reasoning*, pages 65 – 103. Kluwer Academic Publishers, Massachusetts, USA, 1988.

[MacGregor, 1988] Robert MacGregor. A Deductive Pattern Matcher. In *Proceedings of the 1988 Conference on Artificial Intelligence*, St Paul, Mn, August 1988. American Association of Artificial Intelligence.

[Mittal and Paris, 1990] Vibhu O. Mittal and Cécile L. Paris. Analogical explanation in the EES framework. In Nick Filer, editor, *Proceedings of the 5th Workshop on Explanation*, Manchester, UK, April 1990.

[Moore, 1989] Johanna Doris Moore. *A Reactive Approach to Explanation in Expert and Advice-Giving Systems*. PhD thesis, University of California – Los Angeles, 1989.

[Neches *et al.*, 1985] Robert Neches, William Swartout, and Johanna Moore. Enhanced maintenance and explanation of expert systems through explicit models of their development. *IEEE Transactions on Software Engineering*, SE-11(11), 1985.

[Paris, 1989] Cécile Paris. The Use of Explicit User Models in a Generation System for Tailoring Answers to the User's Level of Expertise. In Alfred Kobsa and Wolfgang Wahlster, editors, *User Models in Dialog Systems*. Springer Verlag, Berlin Heidelberg New York Tokyo, 1989.

[Paris, 1990] Cécile L. Paris. Generation and explanation: Building an explanation facility for the explainable expert systems framework. In Cécile Paris, William Swartout, and William Mann, editors, *Selected papers from the 4th International Workshop on Text Generation*. Catalina Island, Kluwer Academic Publishers, 1990. To appear.

[Saha, 1988] P. K. Saha. Metaphorical style as message. In David H. Helman, editor, *Analogical Reasoning*, pages 41 – 65. Kluwer Academic Publishers, Massachusetts, USA, 1988.

[Swartout and Smoliar, 1987] William Swartout and Stephen Smoliar. Explaining the link between causal reasoning and expert behaviour. In *Proceedings of the Symposium on Computer Applications in Medical Care*, Washington, DC., November 1987. Also to appear in "Topics in Medical Artificial Intelligence"; Miller, P.L. (ed), Springer-Verlag.

[The Penman Natural Language Group, 1989] The Penman Natural Language Group. *The Penman Reference Manual*. USC/Information Sciences Institute, Marina del Rey, CA 90282, 1989.